CPSC 3720, Lecture 7: Requirements Analysis: Use Case Analysis

Eileen T. Kraemer and Murali Sitaraman, Clemson
E-mails: etkraem@clemson.edu and murali@Clemson.edu

Spring 2018 version

Some of the lecture slides have benefitted from the findings of grants from the US National Science Foundation.

Quiz

- Quiz #5 on structured analysis

Quiz return

- Quiz #4 on requirements engineering

Requirements Engineering

- Requirements elicitation
- Requirements modeling
- Requirements analysis
- Requirements documentation
- Requirements validation
- Requirements management

Requirements Modeling

- Key Idea: Understand which aspects of the system are relevant to model and which are irrelevant

- Develop a model that is sufficiently complex, but no more so than necessary to capture the essential elements

Analysis Methods

- Structured Analysis
  - Function-oriented analysis
  - (previous lecture)
- Use case analysis
  - Usage-oriented analysis
- Others...

Scenario-based requirements elicitation

- technique of asking questions related to a descriptive story to obtain design requirements

- Scenario
  - subset of a use case
  - sequence of actions that illustrates behavior
  - illustrates an interaction or the execution of a use case instance

Unified Modeling Language (UML)

- a modeling language & diagrammatic notation for object-oriented programming
- Several types of diagrams, including
  - use case diagrams (for requirements)
  - state diagrams (for OO analysis)
  - class diagrams and sequence diagrams (for OO design)
- common means of communication for software engineers

Example scenario

- Example from a Monopoly game:
  - *Player1 lands on Boardwalk, which contains a house owned by Player 2, and the rent is $100. Player1 gives Player2 $100.*

Scenario-based elicitation

- The Software Engineer (SE) queries stakeholders for the kinds of things they want the system to do, how they envision using system
- SE maps this system problem to a system spec represented as a set of actors & use cases
- A complete set of scenarios should describe everything the system should do

Use case

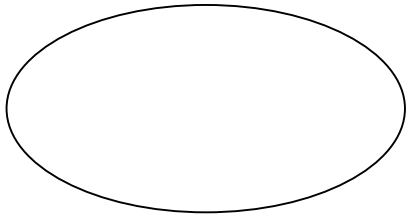
- a specification of a sequence of actions, including variant sequences and error sequences, that a system, subsystem or class can perform by interacting with outside actors
- Use case
  - a set of scenarios, tied together by a common user goal or set of transactions performed by a system that produces an outwardly visible result for an actor

Use case: Get out of Jail

- Set of scenarios:
  - *A player is in Jail. The player clicks the "Get out of Jail" button. $50 is decremented from their money. The player can then roll the dice and continue with the game.*
  - *A player is in Jail. The player clicks the "Get out of Jail" button. The player has less than $50. The player becomes bankrupt and all the tradeable cells he or she owns become available in the game. The player is out of the game.*

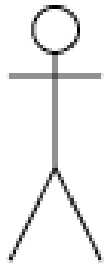
Basic UML symbols

- UML symbol for a use case:



  - labeled with a present-tense verb phrase in active voice: "**Draw** Card", "**Get Out** of Jail", "**Switch** Turn"

Actor



- Interacts with the system
  - supplies input information to the system
  - receives information from the system
  - both supplies input information to and receives information from the system
- but is not part of the system
- represents a **role** rather an individual
  - need not be a person (could be a DB, another system, etc.)

Identifying the actors

- Example questions:
  - Who uses the system?

Identifying the actors

- Example questions:
  - Who uses the system?
  - Who installs the system?

Identifying the actors

- Example questions:
  - Who uses the system?
  - Who installs the system?
  - Who starts up the system?

Identifying the actors

- Example questions:
  - Who uses the system?
  - Who installs the system?
  - Who starts up the system?
  - Who maintains the system?

Identifying the actors

- Example questions:
  - Who uses the system?
  - Who installs the system?
  - Who starts up the system?
  - Who maintains the system?
  - Who shuts down the system?

Identifying the actors

- Example questions:
  - Who uses the system?
  - Who installs the system?
  - Who starts up the system?
  - Who maintains the system?
  - Who shuts down the system?
  - What other systems use this system?

Identifying the actors

- Example questions:
  - Who uses the system?
  - Who installs the system?
  - Who starts up the system?
  - Who maintains the system?
  - Who shuts down the system?
  - What other systems use this system?
  - Who gets information from this system?

Identifying the actors

- Example questions:
  - Who uses the system?
  - Who installs the system?
  - Who starts up the system?
  - Who maintains the system?
  - Who shuts down the system?
  - What other systems use this system?
  - Who gets information from this system?
  - Who provides information to the system?

Identifying the actors

- Example questions:
  - Who uses the system?
  - Who installs the system?
  - Who starts up the system?
  - Who maintains the system?
  - Who shuts down the system?
  - What other systems use this system?
  - Who gets information from this system?
  - Who provides information to the system?
  - Does anything happen automatically?

Identifying the Use Cases

- What outwardly visible, measurable result of value does each actor desire?
- Questions:
  - What functions will the actor want from the system?

Identifying the Use Cases

- What outwardly visible, measurable result of value does each actor desire?
- Questions:
  - What functions will the actor want from the system?
  - Does the system store information? Which actors will create, read, update or delete this information?

Identifying the Use Cases

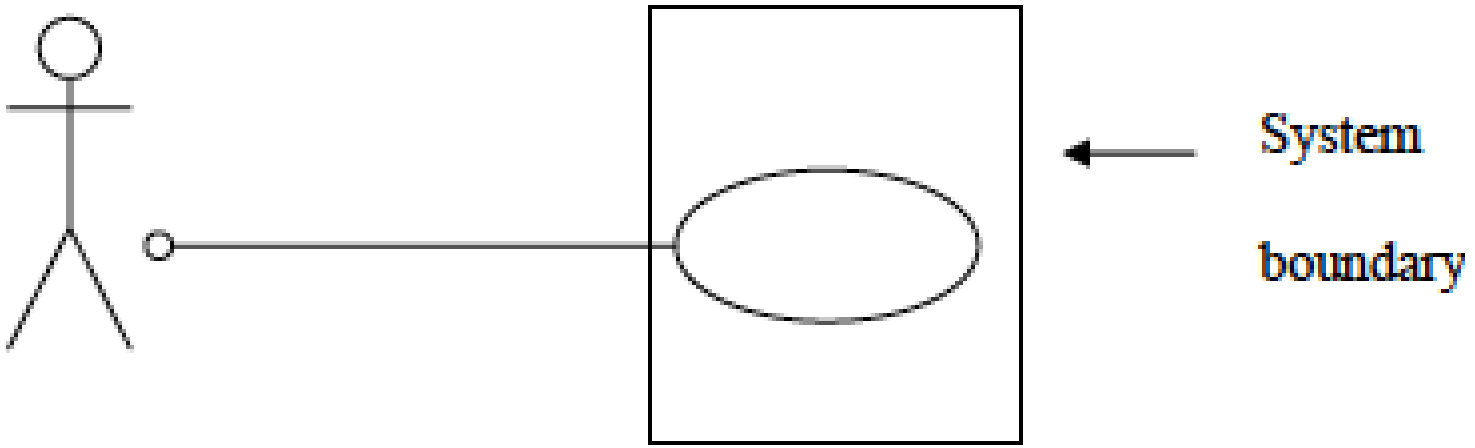
- What outwardly visible, measurable result of value does each actor desire?
- Questions:
  - What functions will the actor want from the system?
  - Does the system store information? Which actors will create, read, update or delete this information?
  - Does the system need to notify an actor about changes in the internal state?

Identifying the Use Cases

- What outwardly visible, measurable result of value does each actor desire?
- Questions:
  - What functions will the actor want from the system?
  - Does the system store information? What actors will create, read, update or delete this information?
  - Does the system need to notify an actor about changes in the internal state?
  - Are there any external events the system must know about? Which actor informs the system of those events?

Identifying the boundary

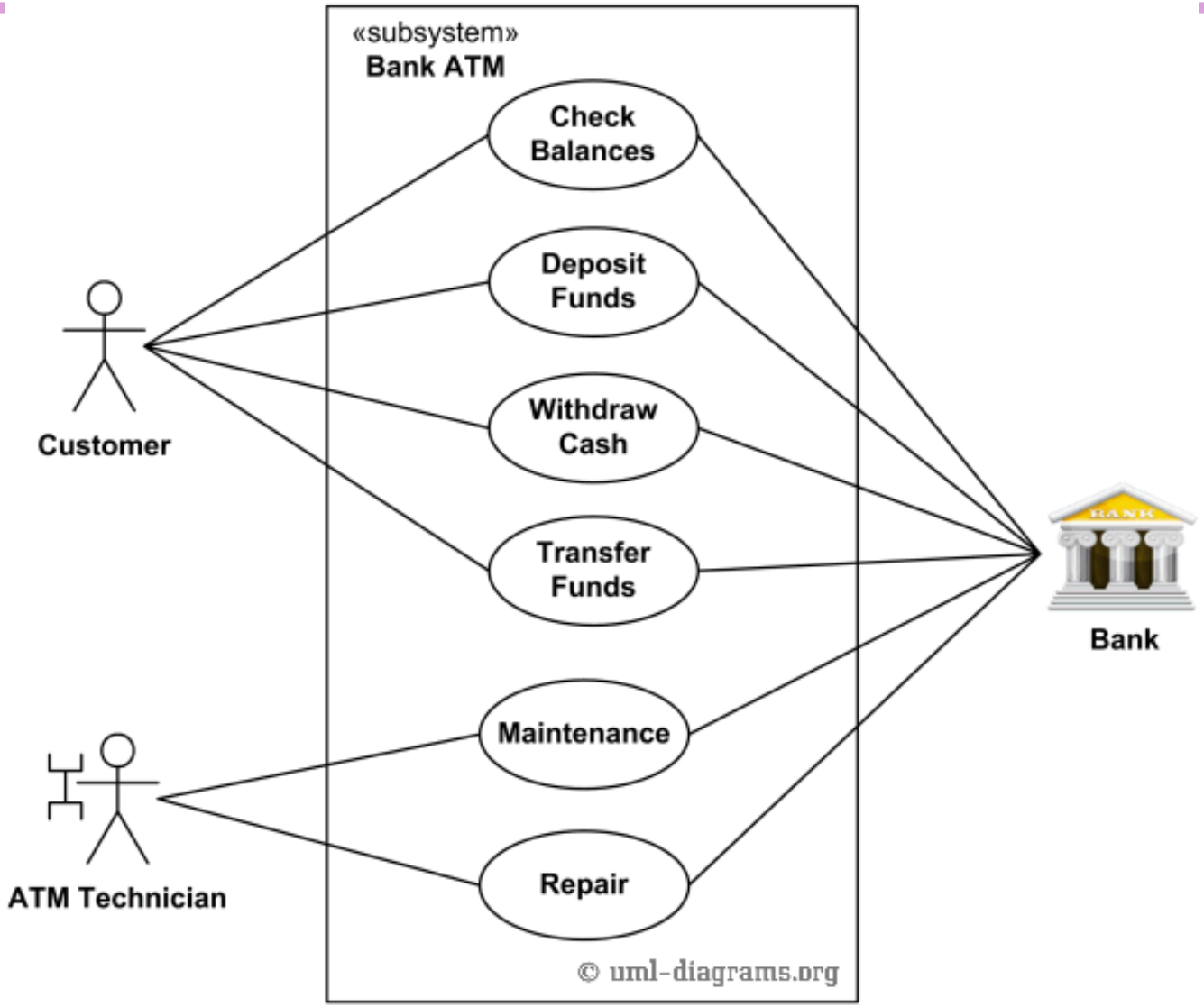
- The boundary separates:
  - things you need to create in your system
  - things outside the system & that you don't need to create



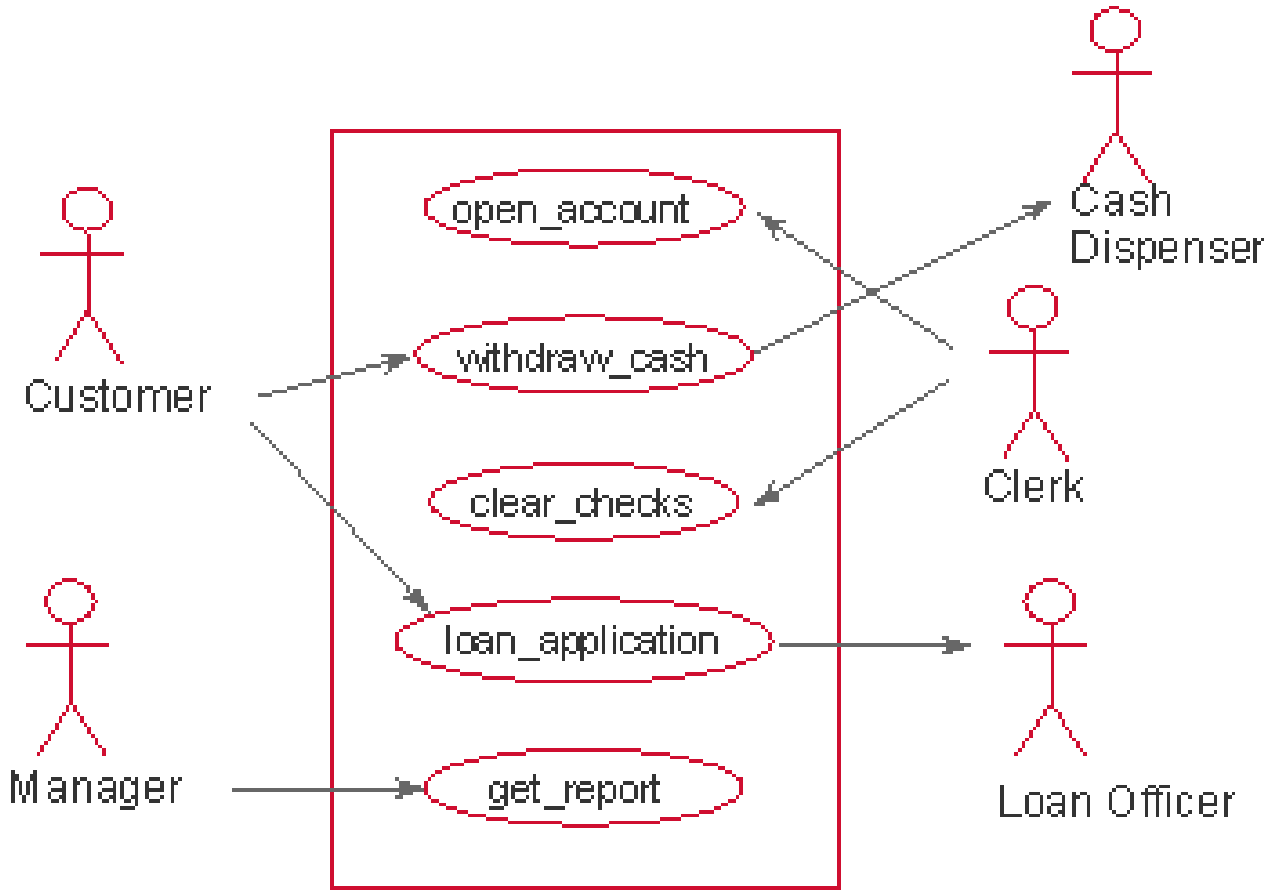
Use Case Diagram

- visual representation of relationships between actors & use cases
- documents system's intended behavior

Simple ATM example



Banking System Use Case Diagram



Use Case Relationships

- communication
- include
- extend

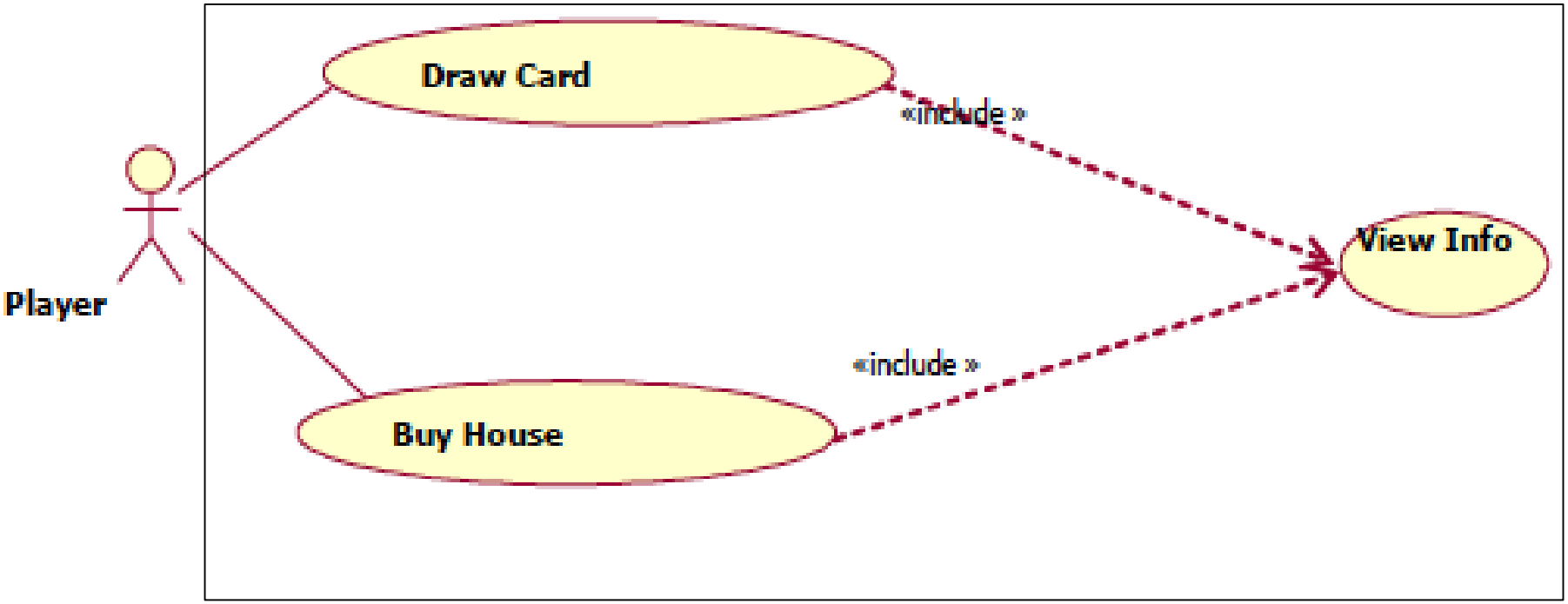
communication relationship

- the default relationship between actors & use cases
- one entity initiates communication or invokes request
  - actor can communicate with use case
  - use case can communicate with use case
- indicated by line with no label or arrow

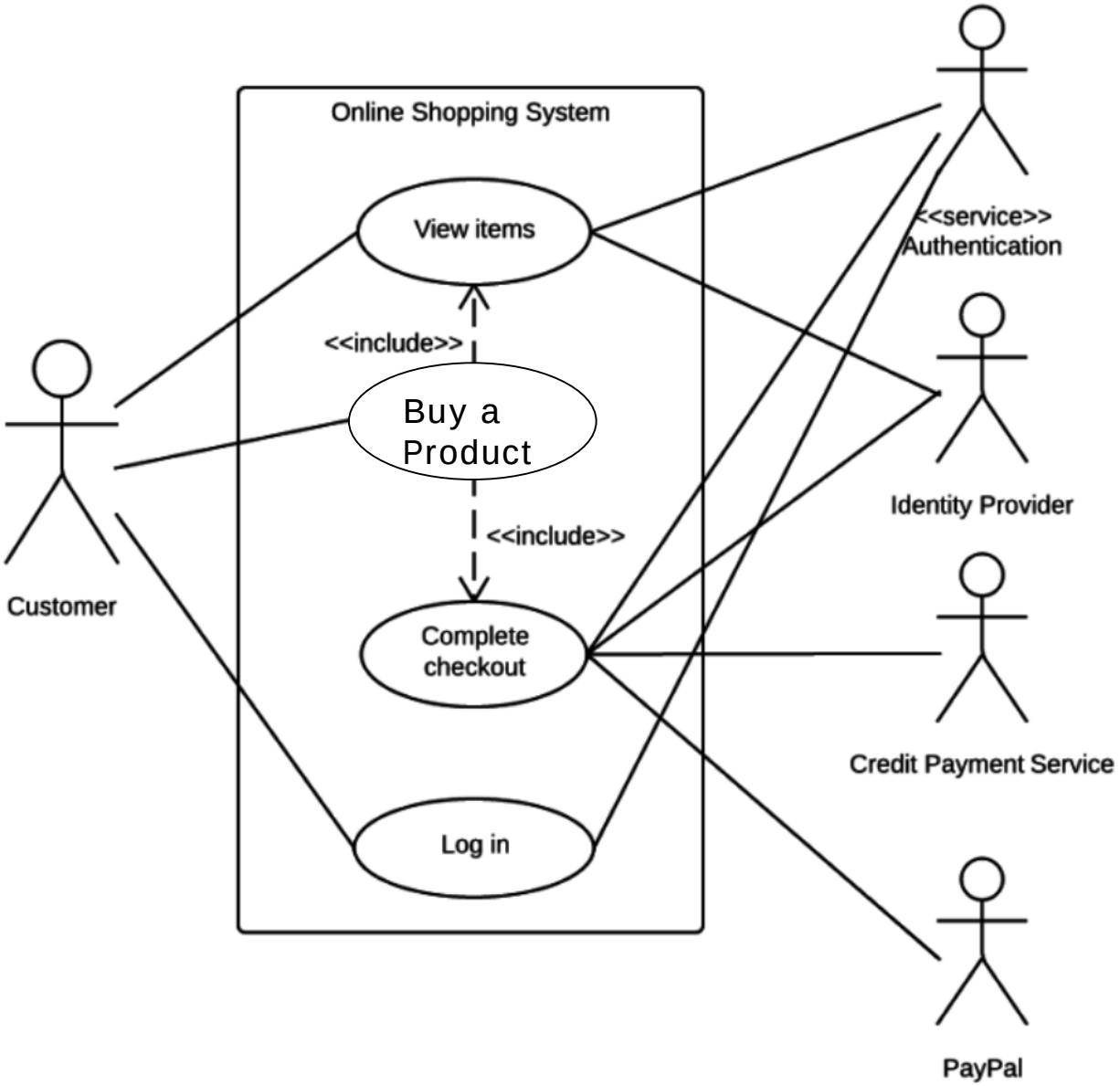
The *include* relationship

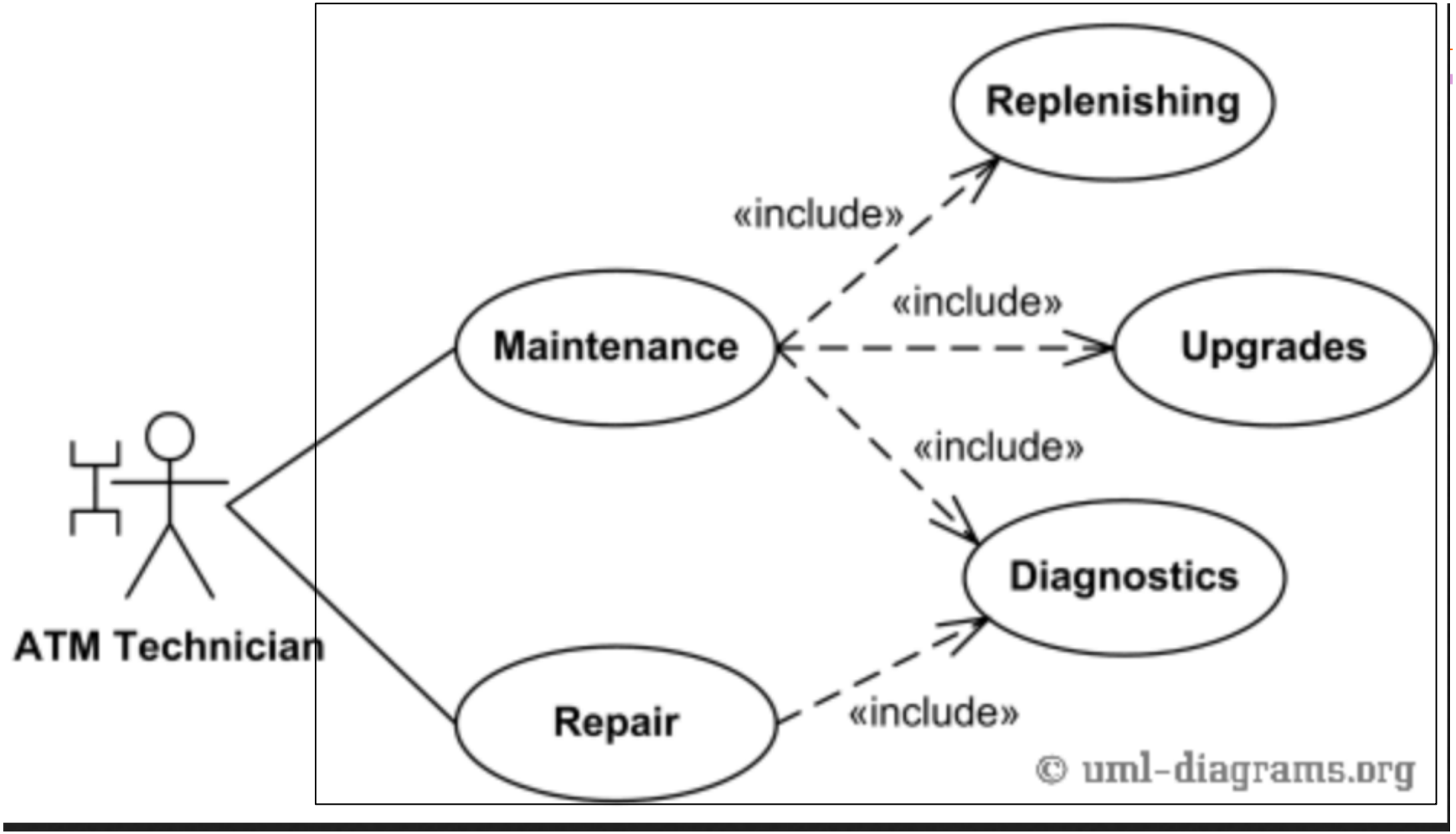
- indicates that one use case is used in another use case's functionality
- employed when a chunk of behavior repeats across use cases
- arrow is labeled with <<include>>

Example using <<include>>



Example


Online Shopping System
View items
<<include>>
Buy a Product
<<include>>
Complete checkout
Log in
Customer
<<service>> Authentication
Identity Provider
Credit Payment Service
PayPal

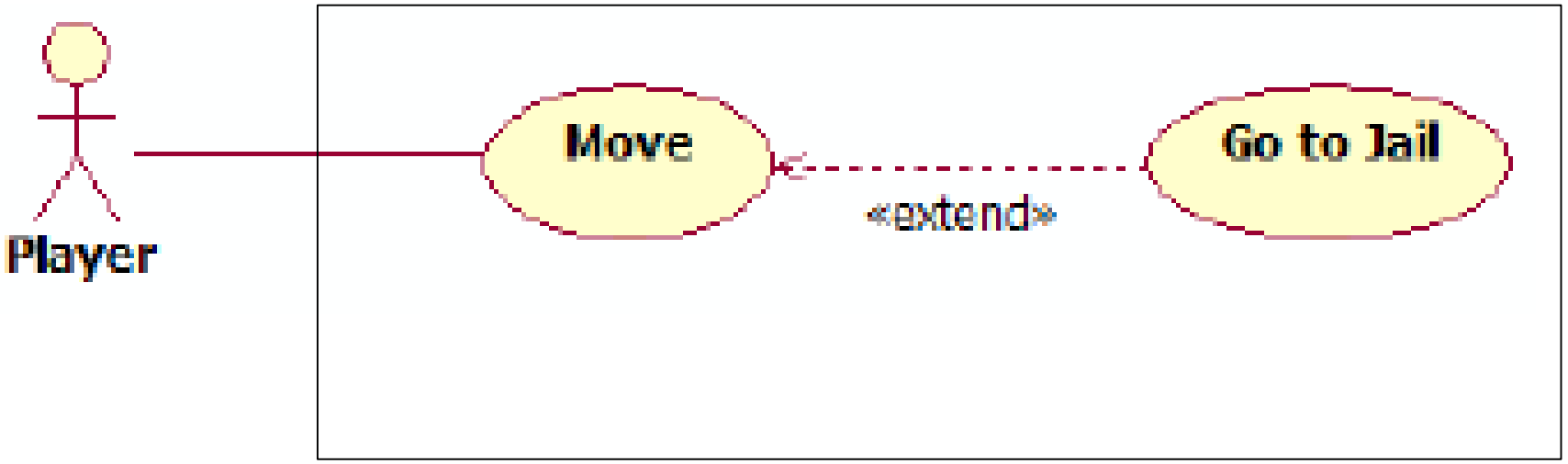

Replenishing
«include»
Maintenance
«include»
Upgrades
«include»
Diagnostics
ATM Technician
Repair
«include»
© uml-diagrams.org

The *extend* relationship

- used when describing a variation on normal behavior, executed only under certain conditions
- similar to alternative flows but used when alternative flow is complex or multi-step
- represented by a dotted line whose arrow points toward a use case that is being extended

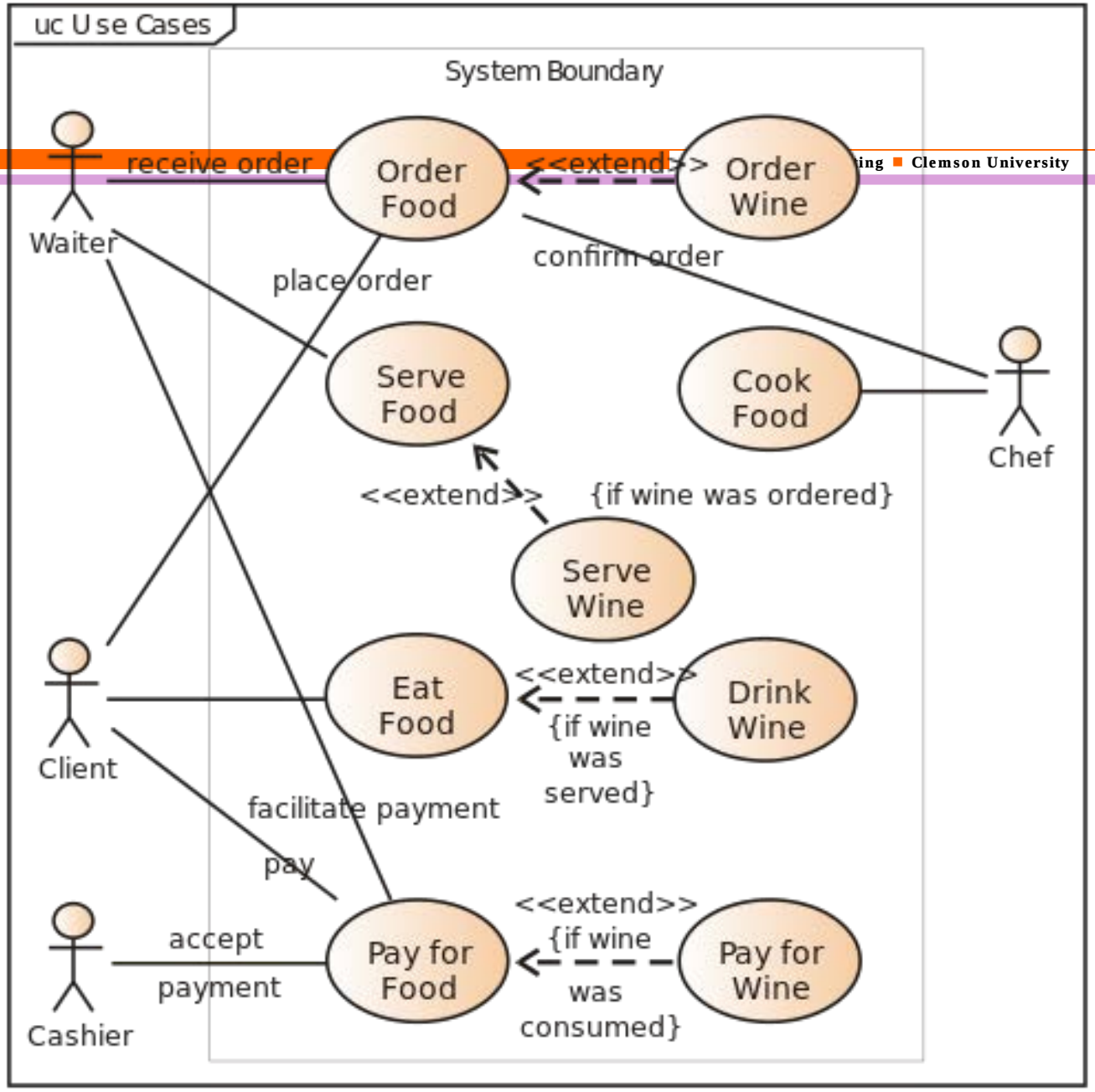
Example using <<extend>>

- Scenario:
  - *A player moves on the board because he or she has to go to Jail.*
  - *A player moves on the board because he or she has to go to Free Parking.*



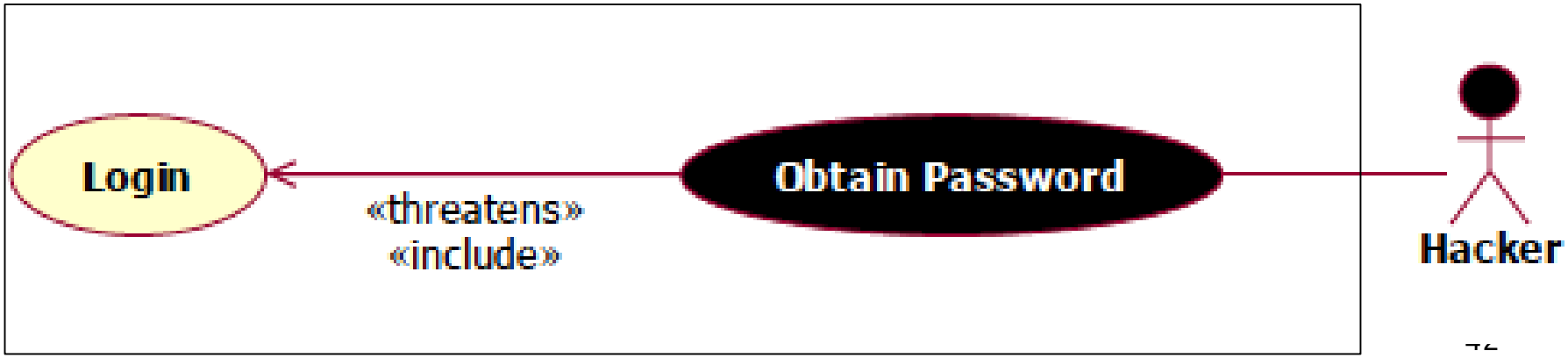
include vs. *extend*

- Use Case X *includes* Use Case Y
  - in the course of doing X (or a subtask of X), Y will **always** be completed
- Use Case X *extends* Use Case Y
  - Y performs a sub-task and X is a similar but more specialized way of accomplishing that subtask
  - **X only happens in an exception situation**
- Note: use <<extend>> sparingly


uc Use Cases
System Boundary
Waiter
receive order
Order Food
<<extend>>
Order Wine
confirm order
place order
Serve Food
Cook Food
Chef
<<extend>>
{if wine was ordered}
Serve Wine
Client
Eat Food
<<extend>>
{if wine was served}
Drink Wine
facilitate payment
pay
Cashier
accept payment
Pay for Food
<<extend>>
{if wine was consumed}
Pay for Wine

Misuse Cases

- a use case from the point of view of a hostile actor
- used to represent privacy/security requirements
- black ellipse identifies misuse case



Text and diagram

- Although the diagrammatic representation of a UML use case is standardized, the format of the associated text is not
  - and text is likely more important …
- We'll look at one format for text:
  - itemized list format (Fowler, UML Distilled text)
- But others exist:
  - a Flow-of-Events format (Jacobson, Williams)

Fowler, UML Distilled

Buy a Product ← **name of use case**

Goal Level: Sea Level

Main Success Scenario:
1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping info (address, delivery speed)
4. System presents full pricing information, including shipping
5. Customer fills in credit information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

Extensions:
3a: Customer is regular customer
    .1 System displays current shipping, pricing and billing information
    .2 Customer may accept or override defaults, return to MSS at step
6a: System fails to authorize credit purchase
    .1 Customer may re-enter credit card info or may cancel

Fowler, UML Distilled

Buy a Product

Goal Level: Sea Level **<- level of abstraction (Kite, Sea, Fish)**

Main Success Scenario:
1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping info (address, delivery speed)
4. System presents full pricing information, including shipping
5. Customer fills in credit information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

Extensions:
3a: Customer is regular customer
    .1 System displays current shipping, pricing and billing information
    .2 Customer may accept or override defaults, return to MSS at step
6a: System fails to authorize credit purchase
    .1 Customer may re-enter credit card info or may cancel

Fowler, UML Distilled

Buy a Product

Goal Level: Sea Level

Main Success Scenario: **<- Choose one scenario as main scenario**

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping info (address, delivery speed)
4. System presents full pricing information, including shipping
5. Customer fills in credit information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

Extensions: **<- Other scenarios in use case as extensions**

3a: Customer is regular customer
    .1 System displays current shipping, pricing and billing information
    .2 Customer may accept or override defaults, return to MSS at step

6a: System fails to authorize credit purchase
    .1 Customer may re-enter credit card info or may cancel

Fowler, UML Distilled

Buy a Product

Goal Level: Sea Level

Each Use case has a primary Actor
in this case, the **Customer**
Activities assigned to Actor or **System**
(Secondary Actors may also be involved)

Main Success Scenario:

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping info (address, delivery speed)
4. System presents full pricing information, including shipping
5. Customer fills in credit information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to Customer

Who is carrying out the step? What is their intent?

Extensions:

3a: Customer is regular customer
.1 System displays current shipping, pricing and billing information
.2 Customer may accept or override defaults, return to MSS at step

6a: System fails to authorize credit purchase
.1 Customer may re-enter credit card info or may cancel

Fowler, UML Distilled

Buy a Product

Goal Level: Sea Level

Main Success Scenario:

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping info (address, delivery speed)
4. System presents full pricing information, including shipping
5. Customer fills in credit information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

Extensions: **<- result in different interactions than MSS**

3a: Customer is regular customer

.1 System displays current shipping, pricing and billing information

.2 Customer may accept or override defaults, return to MSS at step

6a: System fails to authorize credit purchase

.1 Customer may re-enter credit card info or may cancel

Fowler, UML Distilled

Buy a Product

Goal Level: Sea Level

Main Success Scenario:
1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping info (address, delivery speed)
4. System presents full pricing information, including shipping
5. Customer fills in credit information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

Extensions:
3a: Customer is regular customer **<- Step num and variant condition**
.1 System displays current shipping, pricing and billing information
.2 Customer may accept or override defaults, return to MSS at step 6
6a: System fails to authorize credit purchase
.1 Customer may re-enter credit card info or may cancel

Fowler, UML Distilled

Buy a Product

Goal Level: Sea Level

Main Success Scenario:
1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping info (address, delivery speed)
4. System presents full pricing information, including shipping
5. Customer fills in credit information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

Extensions:
3a: Customer is regular customer
.1 System displays current shipping, pricing and billing information
.2 Customer may accept or override defaults, return to MSS at step 6
6a: System fails to authorize credit purchase
.1 Customer may re-enter credit card info or may cancel

Fowler, UML Distilled

Buy a Product

Goal Level: Sea Level

Use underlining to represent includes

Main Success Scenario:

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping info (address, delivery speed)
4. System presents full pricing information, including shipping
5. Customer fills in credit information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

Extensions:

3a: Customer is regular customer
    .1 System displays current shipping, pricing and billing information
    .2 Customer may accept or override defaults, return to MSS at step

6a: System fails to authorize credit purchase
    .1 Customer may re-enter credit card info or may cancel

Fowler, UML Distilled

Buy a Product

Goal Level: Sea Level

Can also add preconditions, guarantees, and trigger

Main Success Scenario:

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping info (address, delivery speed)
4. System presents full pricing information, including shipping
5. Customer fills in credit information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming email to customer

Extensions:

3a: Customer is regular customer
    .1 System displays current shipping, pricing and billing information
    .2 Customer may accept or override defaults, return to MSS at step

6a: System fails to authorize credit purchase
    .1 Customer may re-enter credit card info or may cancel

Summary

- Identify all actors in system
- Identify functionality that actors want from system
- Consider common functions; abstract as «include» use cases
- Avoid «extend» relationship; overly complex
- Use case diagrams help to visualize what the system has to do and the use case flow-of-events gets specific about what the customer wants – the variations and the exceptions.

Glossary (from Rumbaugh, Jacobson et al., 1999)

- Actor
  - An abstraction for entities outside a system, subsystem, or class that interact directly with the system. An actor participates in a use case or coherent set of use cases to accomplish an overall purpose.
- Scenario
  - A sequence of actions that illustrates behavior. A scenario may be used to illustrate an interaction or the execution of a use case instance.
- Use case
  - The specification of sequences of actions, including variant sequences and error sequences, that a system, subsystem, or class can perform by interacting with outside actors.

Example & Activity

- A Requirements interview:
  - https://www.youtube.com/watch?v=MtW4z8cHp6s

- Identify the actors in this system
- Identify the use cases in this system
- Sketch the associated UML use case diagrams
- Using the Fowler notation, write the text for the use case diagrams